PlayStation Operating System

Table of Contents

List of Figures

List of Tables

About This Manual

This manual is the latest release of instructional material relating to the PlayStation® operating system as of Run-Time Library release 4.3. The purpose of this document is to describe the operating system and related fundamentals.

Changes Since Last Release

There have been no substantial changes in this document since its last release.

Related Documentation

This manual should be read in conjunction with the *Library Overview* and *Library Reference*.

Note: The Developer Support Website posts current developments regarding the Libraries and also provides notice of future documentation releases and upgrades.

Manual Structure

Section	Description
Ch. 1: <PS-X OS> Overview	A flexible and powerful operating system is part of the PS-X. By using this operating system, you may maximize the capabilities of PS-X.
Ch. 2: <PS-X OS> in Reality	Describes starting the OS, using the memory map and other activities that one must understand when accessing the operating system.
Ch. 3: <PS-X OS> Service	The most significant characteristic of <PS-X OS> is that all features controlling the R3000 and PS-X hardware are provided as C functions. Because of this, the burden on the program developer is lightened; he or she may proceed with development more efficiently.
Ch. 4: Programming Tips	Describes precautions and tips when programming with <PS-X OS>.

Developer Reference Series

This manual is part of the *Developer Reference Series*, a series of technical reference volumes covering all aspects of PlayStation development. The complete series is listed below:

Manual	Description
PlayStation Hardware	Describes the PlayStation hardware architecture and overviews its subsystems.
PlayStation Operating System	Describes the PlayStation operating system and related programming fundamentals.
Run-Time Library Overview	Describes the structure and purpose of the run-time libraries provided for PlayStation software development.
Run-Time Library Reference	Defines all available PlayStation run-time library functions, macros and structures.
Inline Programming Reference	Describes in-line programming using DMPSX, GTE inline macro and GTE register information.

SDevTC Development Environment	Describes the SDevTC (formerly "Psy-Q") Development Environment for PlayStation software development.
3D Graphics Tools	Describes how to use the PlayStation 3D Graphics Tools, including the animation and material editors.
Sprite Editor	Describes the Sprite Editor tool for creating sprite data and background picture components.
Sound Artist Tool	Provides installation and operation instructions for the DTL-H800 Sound Artist Board and explains how to use the Sound Artist Tool software.
File Formats	Describes all native PlayStation data formats.
Data Conversion Utilities	Describes all available PlayStation data conversion utilities, including both stand-alone and plug-in programs.
CD Emulator	Provides installation and operation instructions for the CD Emulator subsystem and related software.
CD-ROM Generator	Describes how to use the CD-ROM Generator software to write CD-R discs.
Performance Analyzer User Guide	Provides general instructions for using the Performance Analyzer software.
Performance Analyzer Technical Reference	Describes how to measure software performance and interpret the results using the Performance Analyzer.
DTL-H2000 Installation and Operation	Provides installation and operation instructions for the DTL-H2000 Development System.
DTL-H2500/2700 Installation and Operation	Provides installation and operation instructions for the DTL-H2500/H2700 Development Systems.

Typographic Conventions

Certain Typographic Conventions are used through out this manual to clarify the meaning of the text. The following conventions apply to all narrative text except for structure and function descriptions:

Convention	*Meaning*
`courier`	Indicates literal program code.
Bold	Indicates a document, chapter or section title.

The following conventions apply within structure and function descriptions only:

Convention	*Meaning*
Medium Bold	Denotes structure or function types and names.
Italic	Denotes function arguments and structure members.

Developer Support

Sony Computer Entertainment America (SCEA)

SCEA developer support is available to licensees in North America only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
In North America	*In North America*
Attn: Developer Tools Coordinator Sony Computer Entertainment America 919 East Hillsdale Blvd., 2nd floor Foster City, CA 94404 Tel: (650) 655-8000	E-mail: DevTech_Support@playstation.sony.com Web: http://www.scea.sony.com/dev Developer Support Hotline: (650) 655-8181 (Call Monday through Friday, 8 a.m. to 5 p.m., PST/PDT)

Sony Computer Entertainment Europe (SCEE)

SCEE developer support is available to licensees in Europe only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
In Europe	*In Europe*
Attn: Production Coordinator Sony Computer Entertainment Europe Waverley House 7-12 Noel Street London W1V 4HH Tel: +44 (0) 171 447 1600	E-mail: dev_support@playstation.co.uk Web: https://www-s.playstation.co.uk Developer Support Hotline: +44 (0) 171 447 1680 (Call Monday through Friday, 9 a.m. to 6 p.m., GMT or BST/BDT)

Chapter 1:
<PS-X OS> Overview

What is <PS-X OS>

<PS-X OS> was developed for the R3000, the CPU of PS-X. The efficiency of program development is very dependent on the environment and services that the operating system in a machine provides. If it has a fast enough CPU and peripheral devices, there is no need to take time to figure out a method of maximizing the capabilities of the hardware and one may concentrate on programming by skillfully using the services provided by the OS.

The design concept of the <PS-X OS> lies in providing an environment for the developer of game programs and for enabling you to control programs with interrupts more easily. Based on this concept the kernel of the <PS-X OS> is provided as the aggregate of the services (subroutines) controlling the R3000 and the PS-X hardware.

In addition each service is provided as a C language function. By using C, not only may we aim for better source code readability and maintainability, it also makes block structure description and function calls easier and makes it possible to program with greater ease.

Characteristics of <PS-X OS>

There are many characteristics of <PS-X OS> which implement its concept of computing.

Programming with C

For control of the R3000 CPU, control of the PS-X hardware units etc., all services are provided as C language functions. Thus, programming may be performed throughout in C.

Easy Access to the Features of the R3000

Though the interrupt processing procedures in the R3000 are said to be complex, the operating system under the <PS-X OS> makes use of a substitute dispatcher system and provides a simple interface. Of course the overhead that accompanies the dispatcher is kept very low. In addition, in ordinary operating systems the low level implementation details are not disclosed. In this way, operating system chip capabilities are maximized and a higher level of tuning may be achieved. Of course because everything can be done in C, there is no need to learn the intricacies of R3000 assembler.

Light Volume, Small Scale, Emphasis on Performance

If the operating system is slow in speed it could ruin a speedy game program. In order to place emphasis on performance of applications, the size in RAM of <PS-X OS> has been kept to a maximum of 64 Kilobytes, and it was also designed so that the CPU occupied time is minimized. In addition the OS system tables have been disclosed and consideration has been given to expansion of the operating system and acquisition of internal data.

Furthermore, in the <PS-X OS> operating system, checks of prohibited items seen in other operating systems have been eliminated in order to aim for greater speed. These checks should be performed by the application. Thus, though there is the danger that essentially prohibited operations will end up being performed, cautious programming makes possible higher level tuning .

Until now in hardware control of consoles, one had to analyze hardware driver code and painstakingly work through everything with an assembler. In order to lighten this burden all hardware features are provided as C language functions by the <PS-X OS>. The overhead of each function is kept to a minimum.

Single Task or Multi-Tasking

The <PS-X OS> is basically a multi-tasking operating system that can perform multiple processing asynchronously. Multi-task control is appropriate for controlling a CD-ROM, a comparatively slow device, or things like playing background music.

In addition, we have made it possible to select multi-tasking mode as an option for those who are accustomed to traditional programming, in other words, those who are used to single task. Immediately after starting the OS it is single task mode, and it is possible to select operating mode by making a selection at this point.

Device Driver Based File System

The file system (i.e., files of data on CD-ROM) in <PS-X OS> is accessed via a device driver. This allows multiple file systems to coexist and increases development time by avoiding low level file manipulation problems.

Chapter 2:
<PS-X OS> in Reality

Starting and Operating the OS

<PS-X OS> is an operating system that provides an “environment for game program developers.” Thus, it is basically not outfitted with an interface for the user to operate hardware directly (except for the debug monitor in the debugging environment). It is necessary for the application to provide the user interface.

Starting the Operating System

There are two start modes with the <PS-X OS> operating system. The mode is determined by the dip switches of the target box and is changed when the target box is turned on or reset. Please refer to the *PlayStation Hardware* guide for setting the dip switches.

Hardware Mode

This is the mode that corresponds to booting the actual PS-X hardware.

Application Boot

The Application Boot jumps first to a specific address in ROM and performs a check of the hardware connected (CD-ROM drive etc.).

Next, it makes a decision on type of disk in the CD-ROM drive. If the disk is the appropriate one, it retrieves a system designation file, `system.cnf`, and executes it.

If there is no disk it goes into a looping demonstration.

Booting in a PC/AT Compatible Development Environment

When the dipswitch is set so that the system can be booted from a developer host (PC/AT compatible machine), the kernel is set in the default configuration (please refer to the *Run-time Library/Reference*) and then the remote debugger is started. The reading and execution of the execute file that corresponds to the final stage of the boot sequence is performed manually. The kernel configuration cannot be altered.

System Designation File SYSTEM.CNF

The PS-X system settings and the applications that will be started are described here.

The parameters are described from the beginning of a line in `<key word> = <content>` format. Description is all done in 1 byte uppercase alphanumeric characters and a space is necessary on both sides of the equal sign. Lowercase characters may not be used. When the same parameter reoccurs in a file the first version found takes precedence. Below are the parameters that may be described.

Table 2–1: System Designation File Keyword List

Keyword	Content
BOOT	Specify the name of the file to be started. Default is `GAME.PSX`. Example: `BOOT = PSXAPP.EXE`
TCB	Specify the number of tasks. The default is 4 Example: `TCB = 5`
EVENT	Specify the number of events. The default is 16. Example: `EVENT = 5`
STACK	The stack pointer value when the file specified by BOOT is started. Default is 0x8001FF00. Example: `STACK = 800FFF0`

Debug Mode

Debug mode is valid in the debug environment, in other words only with the target box. There are three applications in debug mode.

Debug Monitor

The AT register is used as a work area by the assembler. C compiler and assembler programmer are not permitted to use this register.

SCSI Monitor

SCSI channel 0 is a CPU device, and may be used for execution of remote commands from a host, debugging at C source level, as a DOS device driver and for file transfers.

File System

In the PS-X, CD-ROM is used as the application supply medium. That is to say, the file system of the <PS-X OS> takes as a premise that files are on CD-ROM.

The file system of PS-X is based on ISO 9660 level 1 which is an international standard.

File Name Conventions

File names take the format of `<BASENAME><EXT>`. The `<BASENAME>` is eight 1 byte alphanumeric characters and the `<EXT>` is 3 alphanumeric characters. 1 byte katakana and 2 byte Chinese characters (kanji) cannot be used. `<BASENAME>` and `<EXT>` are separated by a period. This is the same name format as used in MS-DOS.

Subdirectories

Subdirectories cannot be created. Files are all arranged in the root directory.

Number of Files

There is no limit on the number of files that may be placed in the root directory.

Memory Management

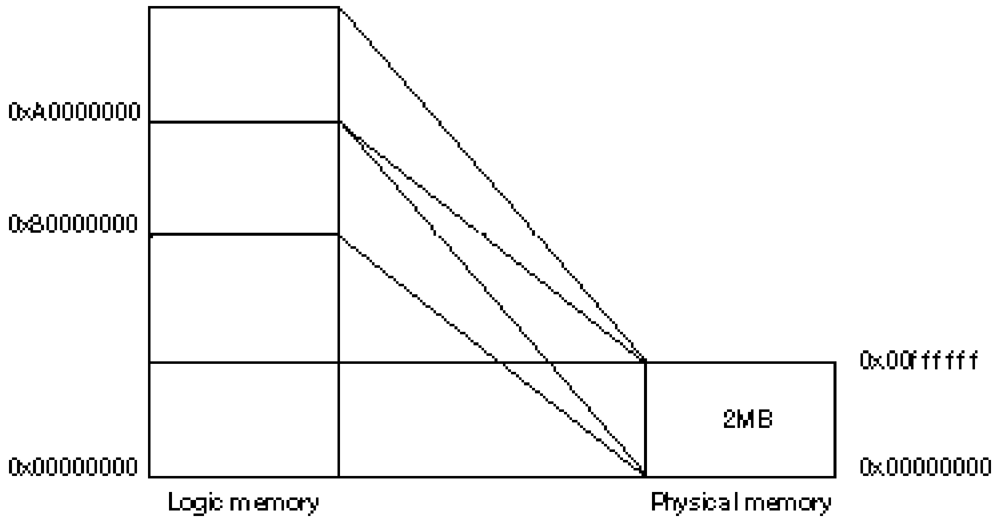
Using memory space on the R3000 is important when using <PS-X OS>. Here we shall explain the memory management of the <PS-X OS>, focusing on the memory map.

Memory Map

We will explain the memory map of <PS-X OS>.

To begin with, we call the space that a program uses logic space. Logic space is expressed by a 32-bit address, and is mapped on physical space in multi layer. In C language it is accessed with a long pointer.

Figure 2–1: Logic Memory/Physical Memory Allocation



Physical space is an address in the installed memory device (i.e., an address on the actual memory chip). If a logic space address is accessed that is not mapped in physical space, a bus error will be generated.

Interface registers with peripheral devices are also mapped in logic space (i.e., memory mapped I/O). Memory devices may be accessed with 1 to 4 byte units but accessing the interface register is done by a method that uses specific size units. When accessed by units other than this a bus error will be generated.

Table 2–2: Memory Map

Logic space (32 bit)	Segment name	Cache
0x00000000~max0x00ffffff	A	Valid
0x1f800000~0x1f8003ff	S	Invalid
0x1f801000~0x1fbfffff	X	Invalid
0x1fc00000~0x1fc7ffff	P	Valid
0x80000000~0x80000000+max	B	Valid
0x9fc00000~0x9fc7ffff	Q	Valid
0xa0000000~0xa0000000+max	C	Invalid
0xbfc00000~0xbfc7ffff	R	Invalid

Physical space	Corresponding logic segment	Device
0x000000~max0x00ffffff	A • B • C	RAM
0x1f800000~0x1f8003ff	S	RAM in CPU
0x1f801000~0x1fbfffff	X	Device
0x1fc00000~0x1fc7ffff	P • Q • R	ROM

I Cache

The validity/invalidity of the Instruction Cache (hereafter I cache) is determined by the upper 4 bits of the logic space. With the segments that correspond to the memory devices mentioned earlier I cache is valid for A, B, P and Q, but is invalid with C and R. Segments in the R3000 refer to block units separating memory space from features and differ in this respect from other CPU's (8086 type), so please be careful.

When I cache is valid, machine code is read into I cache in a block. If the desired instruction is in I cache, main memory does not need to be accessed via the bus. Speed of execution of applications is improved.

The I cache is packaged with 1K words (4 kilobytes), and one way mapped. In other words, logic space is divided into 4 kilobyte units, and these units are mapped onto the I cache.

D Cache

The D cache is configured by the high-speed memory loaded in the CPU. Internally it is structured as a scratch pad and is mapped in the memory space as S segment so that developers can access it.

Both data and programs may be placed in this segment. However, it cannot be the target of a DMA transfer.

System Tables (ToT)

In order to uniformly access each type of system table used internally in the operating system, system table information is disclosed as the structure ToT (Table of Tables). ToT is placed at address 0x00000100.

The contents of ToT are as follows:

Table 2–3: System Tables

Entries	Nature
0	Queue header requires interrupt processing
1	Task state queue header
2	Task management block
3	System reserved
4	Event management block
5-31	System reserved

Definition of ToT Data Structure

The ToT data structure is described by the following C structure.

The structure is defined in the header file `include/kernel.h`.

```
struct ToT {                        /*system table table*/
       unsigned long *head;         /*system table lead address*/
       long size;                   /*system table size (byte)*/
};
```

R3000 Registers

Here we shall explain the R3000 registers. For higher level programming knowledge of the R3000 registers is necessary but in normal C programming there is no need to be especially conscious of it.

The R3000 registers may be divided into General Purpose Registers and a Program Counter.

General Purpose Registers

There are 32, 32-bit general purpose registers. Each register is assigned to the following specific uses by the compiler. When using a threaded data base and developing with an assembler, it is necessary to use the registers in the following way:

Table 2–4: R3000 General Purpose Registers

Macro (1)	Macro (2)	Assignment
R_ZERO	R_R0	0 fixed
R_AT	R_R1	Assembler reserves
R_V0	R_R2	Return value
R_V1	R_R3	Return value (for double type)
R_A0	R_R4	Argument #1
R_A1	R_R5	Argument #2
R_A2	R_R6	Argument #3
R_A3	R_R7	Argument #4
R_T0	R_R8	Temporaries
R_T1	R_R9	Temporaries
R_T2	R_R10	Temporaries
R_T3	R_R11	Temporaries
R_T4	R_R12	Temporaries
R_T5	R_R13	Temporaries
R_T6	R_R14	Temporaries
R_T7	R_R15	Saved temporaries
R_S0	R_R16	Saved temporaries
R_S1	R_R17	Saved temporaries
R_S2	R_R18	Saved temporaries
R_S3	R_R19	Saved temporaries
R_S4	R_R20	Saved temporaries
R_S5	R_R21	Saved temporaries
R_S6	R_R22	Saved temporaries
R_S7	R_R23	Saved temporaries
R_T8	R_R24	Temporaries
R_T9	R_R25	Temporaries
R_K0	R_R26	Kernel reserved #0
R_K1	R_R27	Kernel reserved #1
R_GP	R_R28	Global pointer
R_SP	R_R29	Stack pointer
R_FP	R_R30	Frame pointer
R_RA	R_R31	Return address

Precautions to be taken when using the general purpose registers are as follows:

AT Register

The assembler uses the AT register as a work area. The C compiler and assembler program are prohibited from using this register.

Return Address

The R3000 does not have a subroutine call concept. Therefore, this is substituted by a jump command saving the return address in a register. The assembler may specify the register where it is saved but the compiler is restricted to the RA register.

C Language Function Arguments

When there are less than 4 arguments they are stored in A0 to A3 registers in order from the left. When there are more than 4 arguments they are accumulated on the stack. Space for 4 arguments is allocated on the stack, but is dummy. The remaining arguments are passed on the stack; the first 4 are in registers as usual.

C Language Function Return Value

The return value of a function is stored in the V0 register when less than 32 bits. When 64 bits (double) the 32 high order bits are stored in the V1 register.

Stack

There is no stack concept in the R3000. Therefore, the compiler stores the pointer in the SP register and implements a stack. In order to efficiently use a function frame (a memory area for automatic variables and a work area) it stores the lead address of the frame for that function in the FP register. This value is determined initially by SP. When the module is started, FP is set to SP.

Global Pointer

The R3000 memory is accessed by "encoded 16-bit offset register indirect mode." In order to effectively accomplish this, the compiler consolidates the variables up to 64 Kilobytes in a "bss section" block. Here the block's start address is stored in the GP register and using the above mode access is performed by one word. This address value is called a global pointer and it does not vary in the module.

Program Counter

The program counter cannot be accessed directly.

Immediately after Power-On Reset

The program counter will be 0xbfc00000.

When there is an external interrupt and when an error (except a debug error) occurs

- The content of the program counter will be saved to the EPC register of coprocessor 0 (the section that handles errors and interrupts in the R3000).
- The processor jumps to 0x00000080.

Chapter 3:
<PS-X OS> Service

<PS-X OS> Service

The services provided by <PS-X OS> are called application programmer interfaces (API). The programs that call the OS must have their function libraries linked. All of the PS-X features are covered by the API. Program readability will increase if only API calls are used.

Application Interfaces (API)

A general service for controlling PS-X hardware including the CPU. All of the PS-X features are covered by the API.

The API has the following services. Please refer to the *Run-time Library/Reference* concerning functions provided by the API, and details about data structures.

Module Management Service

Module management is a basic service that loads and executes user application programs.

Event Management Service

Event management is a service controlling program execution triggered by asynchronously occurring events.

Controller Service

This is a service that deals with the controller, which is the PS-X ‘s chief input device. It supports up to 64 controllers which are configured with a maximum of 16 buttons.

I/O Management Service

This is a service supporting low-level input/output to files and logic devices. All file control in PS-X is performed using this service.

Thread Management Service

Threads are aggregates of resources necessary for executing programs. Specifically they are composed of register states, stack areas and CPU states. Multiple threads, multi-tasking and multi-interrupts can be implemented using thread management.

Other Services

There are various other services, for example, setting up the I cache and the R3000 etc., and authorizing or prohibiting processing of interrupts.

Standard C Library

Provides standard C functions based on K & R such as character string operations, standard input and outputs.

Root Counter Management Service

Provides a counting feature which is used by game programmers for time restrictions and timing adjustments.

A Root counter causes automatic count timing. There are 4 elements to this:

1. Display pixels
2. Horizontal synchronization
3. System clock
4. Vertical synchronization

In all of these counters except vertical synchronization a 16-bit target value can be set.

Counters count from zero and when they reach the target value generate an interrupt, then automatically clear to zero and begin counting again. The target value of vertical synchronization is fixed at 1.

PS-X Library

This is a group of C functions which are for controlling expression features like graphics and sound. It provides the following library files. Please refer to the *Run-time Library/Reference* for details.

Table 3–1: Types of Library Files

Name of library	Feature
libgpu	CPU basic operation
libgte	GTE basic operation
libgs	Expanded graphics system
libpress	Data compression
libstr	Streaming
libsnd	Sound
libapi	Kernel service
libetc	Other

File Format

This defines the internal structure of the files that the PS-X library processes. PS-X special tools also generate and process files in this format.

Chapter 4: Programming Tips

Use in Single-Task Mode

<PS-X OS> is basically a multi-tasking operating system which starts out in single task mode. If the programmer does not start multiple threads it will operate in single task mode. One does not necessarily have to use the multi-tasking feature.

Prohibition of Interrupts

By calling the EnterCritical Section( ) function, interrupts can be stopped. Authorization is performed by EnterCritical Section( ).

Overlay Processing

Overlay processing in the PS-X program takes the form of executing a child process from a resident module.

The operating system drives the hardware and starts the file to be executed, but in overlay processing it is necessary for the application to do this.

Character String Display

There is no character generator and fonts are not provided in PS-X. To display character strings the application must provide data and must perform this as part of a graphics operation.

Storage of Game Data

A special PS-X back up cartridge is provided as a game data storage device. Since special functions are also provided for the cartridge, this may be used during programming.

Acceleration of Execution Speed

Here we shall provide a number of tips for increasing the speed of execution of applications.

File Partition and Placement

By partitioning files into appropriate sizes and placing them carefully on the CD-ROM, the speed of starting and reading of data can be increased.

Using the I Cache

Of the segments that correspond to the memory devices described in *Chapter 2, <PS-X OS> in Reality*, I cache is valid with A, B, P and Q and is invalid with C and R. Since the size of I cache is 4 kilobytes, it is very important where and how code is placed. Careful placement of code can result in increased speed.

Using the S Segment

Using the S segment that is part of the CPU’s high-speed memory is another factor in increasing speed.

Heap Memory

The C language heap memory management function group is provided as one part of the C library but <PS-X OS> does not do the initialization necessary for its use. That is because the game program developer must manage all memory available.

When using the heap memory management functions, first execute the heap memory initialization functions belonging to “other services” and then specify the start address and size of the heap memory.

Program Entry Point

The first program started in running PS-X is prepared as an execute file. The file name may be specified following the boot sequence (please refer to the *PlayStation Hardware* manual). The first address to be executed of the execute file—the entry point—may be specified by a function name at link time. When not specified, main( ), the C language default, will be selected. The function that will be the entry point cannot be given an argument.

Vertical Blanking-Period Detection

If the root counter and the event management service are used, the beginning of the vertical blanking period can be detected. The beginning of the vertical blank acts as a trigger so that polling, calling of queue functions and starting of handler functions can be performed. Call queuing functions in the following way:

```
Sample()
{
       unsigned long EV;
       EnterCriticalSection();
       EV = OpenEvent (RCntCNT3, EvSpINT, EvMdNOINTR, NULL);
       EnableEvent (EV);
       SetRCnt (RCntCNT3, 1, RCntMdINTR);
       StartRCnt (RCntCNT3);
       ........
       /* Initialization of controller */
       ........
       while (1){
              WaitEvent (EV);
              ........
       }
}
```

Standard Input/Output

Immediately after starting, standard output is allocated to file descriptor 0 and standard input to file descriptor 1. The input output device is RS232C channel 0 (device name “tty00”). Input output functions such as printf( ) and getchar( ) may be used. The circuit designation of this port is 8-bit characters, 1 stop bit, no parity, hardware flow control. Control with PC/AT can be performed with the special cable packaged together with the target box. A 25 pin type host may be connected by cable.

CD-ROM Simulator

The CD-ROM simulator feature, which uses the developer host file system, is provided as one aspect of the development environment. Please refer to the *SDevTC Development Environment* manual.

Settings Necessary when Activating a Program

In addition to designating a starting address (the program counter initial value) a stack pointer (SP) value should be set. When executing a program and a value corresponding to an address that cannot be accessed is in the stack pointer a bus error will be generated.